REVISTA ODIGOS

Período junio - septiembre 2023
Vol. 4, Núm. 2
rodigos@uisrael.edu.ec
https://doi.org/10.35290/ro.v4n2.2023.899 • e-ISSN: 2697-3405

Mejora de la calidad del software a través de la integración y entrega continua

Improving software quality through continuous integration and delivery

Fecha de recepción: 2023-03-17 • Fecha de aceptación: 2023-04-26 • Fecha de publicación: 2023-06-10

Fabián Lizardo Caicedo Goyes
Universidad Técnica Luis Vargas Torres, Ecuador
fabiancaicedogoyes@hotmail.com
https://orcid.org/0000-0001-5572-6309

RESUMEN

En la actualidad existen muchas formas y métodos de desarrollo de software, los cuales pueden cubrir las necesidades para los que fueron creados o se van a crear, dejando de lado situaciones fundamentales y primordiales a tener en cuenta siempre, como lo son los procesos de seguridad y auditoría de la información. Ahora bien, cuando existen empresas públicas y/o privadas, dedicadas a desarrollar software de manera colaborativa, siempre se debe de tener en cuenta la calidad, medida por la eficiencia y eficacia, tanto en el producto de software, como en su proceso de construcción.

El presente artículo científico tiene como objetivo generar un enfoque para mejorar la calidad del software a través de la implementación de prácticas de integración y entrega continua. Se describen los beneficios de la integración y la entrega continua en el proceso de desarrollo de software, y se proporcionan recomendaciones sobre cómo implementar estas prácticas de manera efectiva.

A su vez, se presentan varios casos de estudio que ilustran cómo la integración continua y la entrega

continua han mejorado la calidad del software en diferentes contextos de desarrollo. Como parte final, se concluye que la estos son prácticas esenciales para mejorar la calidad del software, reducir los costos de desarrollo y aumentar la satisfacción del cliente. Se recomienda que las organizaciones adopten estas prácticas y las integren en su proceso de desarrollo de software.

PALABRAS CLAVE: estrategia de desarrollo, formación combinada, calidad, software, automatización de pruebas

ABSTRACT

At present there are many forms and methods of software development, which can meet the needs for which they were created or will be created, leaving aside fundamental and fundamental situations to always take into account, such as security processes and information auditing. Now, when there are public and/or private companies, dedicated to develop software in a collaborative way, quality must always be taken into account, measured by efficiency and efficacy, both in the software product and in its construction process.

This scientific paper aims to generate an approach to improve software quality through the implementation of continuous integration and delivery practices. The benefits of integration and continuous delivery in the software development process are described, and recommendations are provided on how to implement these practices effectively.

In turn, several case studies are presented that illustrate how continuous integration and continuous delivery have improved software quality in different development contexts. As a final part, it is concluded that these are essential practices to improve software quality, reduce development costs and increase customer satisfaction. It is recommended that organizations adopt these practices and integrate them into their software development process.

KEYWORDS: development strategy, blended learning, quality, software, test automation

Introducción

En la actualidad, la calidad es un factor crítico en el éxito de cualquier proyecto de desarrollo de *software*, la falta o no presencia de ella puede afectar la satisfacción del cliente, la reputación de la empresa y, en última instancia, la rentabilidad del proyecto.

Dentro del desarrollo de *software* es importante tener en cuenta que este pase por un proceso de calidad, es decir, que cada una de sus etapas sea generada de manera eficaz y eficiente. Es importante seguir buenas prácticas de ingeniería de *software*, integrado al uso de metodologías de desarrollo de *software* bien definidas, realización de pruebas de *software* rigurosas, la revisión del código y la arquitectura del *software*, la documentación adecuada, el seguimiento y control de los procesos de desarrollo de *software*, entre otras.

La calidad en el desarrollo de *software* es un término utilizado para medir las características y funcionalidades de un sistema y cómo estas responden en función al tiempo y exactitud. Esto sirve para determinar si un *software* satisface las necesidades y expectativas de sus usuarios y/o clientes. Un *software* de alta calidad debe ser funcional, confiable, seguro, eficiente, fácil de usar y mantener, y debe cumplir con los requisitos establecidos para su desarrollo.

Desde el punto de vista de la ingeniería, la correcta aplicación del ciclo de vida de desarrollo de *software* con herramientas versátiles de apoyo puede dar en cierta medida un índice alto de cómo conseguir un proceso de desarrollo de *software* y un producto de *software* de calidad. Buscando en primera instancia, tener un proceso de planificación de *software* adecuado con herramientas ágiles, conseguir un análisis, diseño y desarrollo veloz con herramientas consolidadas.

La integración continua se refiere a la práctica de fusionar el código fuente de manera regular y automatizada en un repositorio compartido. Esto implica que cada vez que se realizan cambios en el código, se deben ejecutar pruebas automatizadas para detectar errores de inmediato. De esta manera, los errores se pueden corregir rápidamente antes de que se conviertan en problemas mayores y, en última instancia, aumentar la calidad del *software.*

La entrega continua, por su parte, hace referencia a la práctica de liberar el *software* de manera regular y automatizada a través de un proceso de entrega automatizado. Esto significa que cada vez que se realiza una integración exitosa, el *software* se compila, prueba y se implementa automáticamente en un entorno de producción.

Al implementar estas prácticas, los equipos de desarrollo pueden reducir el tiempo de entrega y mejorar la calidad del *software*, al mismo tiempo que reducen los errores y el tiempo que se necesita para solucionarlos. Además, el proceso automatizado permite a los equipos de desarrollo centrarse en tareas más importantes, como la creación de nuevas características y mejoras.

Ahora bien, para conseguir desarrollar *software* de calidad la integración y entrega continua (CI/CD, por sus siglas en inglés) es una práctica de desarrollo de *software* que busca mejorar su calidad y acelerar el proceso de entrega al cliente (Pressman, 2014).

En los últimos años, la integración y la entrega continua han surgido como prácticas clave para mejorar la calidad del *software*. La integración continua se refiere a la práctica de integrar regularmente el código en un repositorio central y ejecutar pruebas automatizadas para detectar errores tempranos en el ciclo de vida del desarrollo del *software* (Fowler, 2006). La entrega continua implica la entrega frecuente de nuevas funcionalidades y correcciones de errores a los usuarios finales, lo que acelera el ciclo de retroalimentación y mejora la calidad del *software* (Humble & Farley, 2011)

Este estudio presenta un enfoque para mejorar la calidad del *software* a través de la implementación de prácticas de integración continua y entrega continua. Se describen los beneficios de estas prácticas y se proporcionan recomendaciones sobre cómo implementarlas de manera efectiva. Se presentan varios casos de estudio que ilustran cómo la integración continua y la entrega continua han mejorado la calidad del *software* en diferentes contextos de desarrollo. Además, se discuten las mejores prácticas para implementar la integración y la entrega continua, incluyendo la automatización de pruebas, la monitorización continua y la colaboración entre equipos de desarrollo y operaciones. Se concluye que ambas son prácticas esenciales para mejorar la calidad del *software* y se recomienda encarecidamente que las organizaciones adopten estas prácticas y las integren en su proceso de desarrollo de *software.*

Metodología

2.1 Marco metodológico

Para llevar a cabo la investigación sobre la mejora de la calidad en el proceso de desarrollo del *software* a través de la integración y entrega continua se optó por una revisión bibliográfica y sobre casos de éxitos.

A continuación de describen cada una de estas:

- **Revisión bibliográfica:** se debe realizar una revisión bibliográfica exhaustiva para comprender los fundamentos teóricos de la integración continua y la entrega continua y su impacto en la calidad del *software*. Dicha revisión puede incluir la lectura de artículos científicos, libros y documentos técnicos.
- **Selección de casos de estudio**: una vez que se han comprendido los fundamentos teóricos, se deben seleccionar varios casos de estudio que ilustren cómo la integración y la entrega continua han mejorado la calidad del *software* en diferentes contextos de desarrollo.
- **Análisis de los casos de estudio:** se deben analizar los casos de estudio seleccionados para identificar los beneficios concretos de la integración continua y la entrega continua en la calidad del *software*. Este análisis puede incluir la revisión de datos cuantitativos y cualitativos, como el número de errores detectados, la frecuencia de lanzamientos, el tiempo de respuesta a los problemas y la satisfacción del cliente.
- **Identificación de mejores prácticas:** a partir del análisis de los casos de estudio, se deben identificar las mejores prácticas para implementar la integración continua y la

entrega continua. Estas mejores prácticas pueden incluir la automatización de pruebas, la monitorización continua y la colaboración entre equipos de desarrollo y operaciones.

- **Validación de los resultados:** finalmente, se deben validar los resultados de la investigación a través de pruebas adicionales o encuestas a los usuarios finales para determinar la eficacia de la integración continua y la entrega continua en la mejora de la calidad del *software.*

Algunos casos de estudios reales encontrados, los cuales han utilizado metodologías para mejorar la calidad del *software* a través de la integración y entrega continua han sido los siguientes que se detallan en la *Tabla 1.*

Tabla 1

Estudios Analizados

Empresa	Detalle	Tecnología
Netflix	Implementó la integración y entrega continua para mejorar la calidad de su *software* y reducir los tiempos de entrega. Con la ayuda de la plataforma de automatización de Jenkins, la empresa pudo automatizar las pruebas y el despliegue de código, lo que permitió la liberación de nuevas características de manera rápida y segura.	Jenkins
Etsy	Es una plataforma de comercio electrónico, la cual implementó la integración y entrega continua para mejorar la calidad de su *software* y reducir los tiempos de entrega. La empresa utilizó una combinación de herramientas de automatización de pruebas y despliegue, como Jenkins, Travis CI y Docker, para reducir el tiempo de lanzamiento de nuevas características de dos semanas a unas pocas horas.	Jenkins, Travis CI y Docker,
Amazon	Amazon utiliza la integración y entrega continua para asegurar la calidad de su *software* en todas sus plataformas y servicios. La empresa utiliza una combinación de herramientas de automatización de pruebas para garantizar que su *software* funcione correctamente en todos los dispositivos y navegadores.	Selenium y Appium
Google	Utiliza la integración y entrega continua para asegurar la calidad de su *software* y reducir los tiempos de entrega. La empresa utiliza una combinación de herramientas de automatización de pruebas y despliegue, como Jenkins y Kubernetes, para garantizar que su *software* funcione correctamente en todos los entornos.	Jenkins y Kubernetes

2.2 Marco conceptual

La mejora de la calidad del *software* a través de la integración y entrega continua se basa en la implementación de prácticas de integración continua (CI) y entrega continua (CD) en el proceso de desarrollo de *software.* A continuación, se presentan los principales conceptos teóricos relacionados con CI/CD y su impacto en la calidad del *software*.

Integración continua: se trata de una práctica de desarrollo de *software* que implica integrar el código de forma regular y frecuente en un repositorio central y ejecutar pruebas automatizadas para detectar errores temprano en el ciclo de vida del desarrollo de *software*. La integración continua ayuda a los equipos de desarrollo a detectar y resolver errores más rápidamente, lo que a su vez mejora la calidad del *software* y reduce los costos de mantenimiento (Rossel, 2017).

Entrega continua: es una práctica de desarrollo de *software* que implica la entrega frecuente de nuevas funcionalidades y correcciones de errores a los usuarios finales. La entrega continua ayuda a los equipos de desarrollo a acelerar el ciclo de retroalimentación y mejora la calidad del *software* al recibir comentarios más rápidamente de los usuarios finales (Humble & Farley, 2011).

Automatización de pruebas: esta es una práctica esencial para la implementación exitosa de CI/CD. Las pruebas automatizadas permiten a los equipos de desarrollo detectar errores temprano en el proceso de desarrollo de *software* y asegurar que este cumpla con los requisitos del usuario y funcione de manera eficiente y confiable (Martin, 2009).

Monitorización continua: es una práctica que implica el seguimiento constante del *software* en producción para detectar errores y fallos. La monitorización continua permite a los equipos de desarrollo detectar problemas en tiempo real y tomar medidas para resolverlos antes de que afecten a los usuarios finales (Hossain & Muhammad, 2017).

Colaboración entre equipos de desarrollo y operaciones: la colaboración entre los equipos de desarrollo y operaciones es esencial para la implementación exitosa de CI/CD. Ambos equipos deben trabajar juntos para asegurar que el *software* se integre y entregue de manera fluida y eficiente (Kim et al., 2016).

2.3 Marco contextual

La mejora de la calidad del *software* a través de la integración y entrega continua es un enfoque que busca automatizar el proceso de construcción, prueba y despliegue de este para garantizar una mayor calidad del producto final. Esta práctica ha sido adoptada por muchas empresas de *software* y se ha convertido en una parte importante del proceso de desarrollo de *software* moderno.

El enfoque de la integración y entrega continua se basa en la idea de que los desarrolladores deben integrar su código en un repositorio central de manera frecuente y regular para detectar problemas tempranos en el ciclo de desarrollo y corregirlos rápidamente. Además, la entrega continua busca automatizar el proceso de construcción, prueba y despliegue de *software* para garantizar que este se entregue de manera rápida y confiable a los usuarios finales.

A su vez, estas se han convertido en un tema de investigación activo en la academia y la industria. Los investigadores y las empresas están buscando formas de mejorar y optimizar el proceso de integración y entrega continua para lograr una mayor calidad del software y una entrega más rápida y confiable.

Además, ambas prácticas se han convertido en una actividad importante en el contexto de DevOps, un enfoque que busca una mayor colaboración entre los equipos de desarrollo y operaciones para lograr una entrega más rápida y confiable de *softwar*e. Asimismo, son fundamentales para la implementación exitosa de DevOps y para lograr los objetivos de entrega continua y mejora de la calidad del *software.*

Resultados

Los resultados de la investigación sobre la mejora de la calidad del *software* a través de la integración y entrega continua han demostrado que esta práctica puede mejorar significativamente su calidad y reducir el tiempo de entrega del producto final. Algunos de los más significativos incluyen:

a. **Reducción de errores y problemas en el *software*:** la integración y entrega continua permiten detectar errores y problemas en él de manera temprana, lo que reduce la probabilidad de que estos errores afecten a los usuarios finales.
b. **Aumento de la eficiencia y productividad:** la automatización de los procesos de integración y entrega continua reduce la necesidad de intervención manual, lo que permite a los desarrolladores centrarse en otras tareas críticas y aumentar su productividad.
c. **Mayor colaboración y transparencia:** la integración y entrega continua fomentan la colaboración entre los miembros del equipo de desarrollo y garantizan que todos estén al tanto de los cambios y actualizaciones realizados en el *software*.
d. **Mejora de la experiencia del usuario:** la entrega continua permite a los usuarios recibir nuevas funcionalidades y mejoras de manera más rápida y confiable, lo que mejora su experiencia general con el *software*.
e. **Reducción del tiempo de lanzamiento:** la entrega continua permite a las empresas lanzar nuevas versiones de *software* con mayor rapidez y frecuencia, lo que les permite mantenerse a la vanguardia de la competencia.

Dentro de las diferentes metodologías de desarrollo de *software* que toman su orientación para mejorar su calidad a través de la integración y entrega continua (CI/CD) se tienen las siguientes:

- **Agile:** el enfoque ágil es una metodología de desarrollo de *software* que se centra en la entrega rápida y continua de *software* de alta calidad. El enfoque ágil utiliza iteraciones cortas y frecuentes para desarrollar y entregar el *software,* lo que permite a los desarrolladores corregir los errores y mejorar la calidad de este de forma continua (Beck et al., 2001).
- **Scrum:** es una metodología ágil que se enfoca en la colaboración y el trabajo en equipo. En Scrum, el equipo trabaja en sprints, que son iteraciones cortas y enfocadas en la entrega de *software* funcional. Asimismo, utiliza reuniones diarias y otras prácticas de colaboración para mejorar la calidad del *software* y acelerar la entrega (Schwaber & Sutherland, 2017).
- **Kanban:** es una metodología ágil que se enfoca en la visualización y optimización del flujo de trabajo. Kanban utiliza tableros visuales para representar el flujo de trabajo y la entrega de *software*. También se enfoca en la entrega continua de *software* de alta calidad y en la eliminación de cuellos de botella y desperdicio en el proceso de desarrollo (Anderson, 2010).

- **DevOps:** como se mencionó anteriormente, DevOps es una metodología que se enfoca en la colaboración entre los equipos de desarrollo y operaciones para lograr una entrega continua de *software* de alta calidad. DevOps utiliza la integración continua, la entrega continua y la automatización para mejorar la eficiencia y calidad del proceso de entrega de *software* (Kim et al., 2016).

A continuación, en la *Tabla 2* se presenta un cuadro comparativo entre algunas de las metodologías de desarrollo de *software* más utilizadas para mejorar su calidad a través de la integración y entrega continua.

Tabla 2

Cuadro Comparativo Metodologías de Desarrollo de Software

Metodología	Enfoque	Características	Ventajas
Agile	Entrega rápida y continua de *software* de alta calidad	Iteraciones cortas y frecuentes, trabajo en equipo, colaboración con el cliente, enfoque en la calidad del *software*, adaptabilidad al cambio	Mejora la eficiencia, calidad y velocidad del proceso de entrega de *software*, mayor satisfacción del cliente
Scrum	Colaboración y trabajo en equipo	Trabajo en sprints, reuniones diarias, colaboración con el cliente, enfoque en la entrega de *software* funcional	Mayor eficiencia y calidad en la entrega de *software*, enfoque en el cliente
Kanban	Visualización y optimización del flujo de trabajo	Tableros visuales, enfoque en la entrega continua de *software* de alta calidad, eliminación de cuellos de botella y desperdicio en el proceso de desarrollo	Mejora la eficiencia y calidad del proceso de entrega de *software,* mayor transparencia en el proceso de desarrollo
DevOps	Colaboración entre los equipos de desarrollo y operaciones	Integración continua, entrega continua, automatización, colaboración entre los equipos de desarrollo y operaciones	Mejora la eficiencia, calidad y velocidad del proceso de entrega de *software*, mayor colaboración y responsabilidad compartida entre los equipos de desarrollo y operaciones

Cada metodología tiene sus propias ventajas y desventajas, y la elección dependerá de las necesidades específicas de la organización y de su entorno de desarrollo de *software.* Agile, Scrum y Kanban se enfocan en la entrega rápida y continua de *software* de alta calidad, mientras que DevOps se enfoca en la colaboración entre los equipos de desarrollo y operaciones para lograr una entrega continua de *software* de alta calidad.

La elección de una metodología dependerá de factores como el tamaño y la complejidad del proyecto, la cultura organizacional, las habilidades y la experiencia del equipo de desarrollo, los recursos tecnológicos disponibles, las expectativas del cliente y otros factores específicos de cada organización. Esta última debe evaluar cuidadosamente estas variables para determinar cuál de las metodologías de desarrollo de *softwar*e es la más adecuada para sus necesidades y objetivos.

Conclusiones

En conclusión, la investigación realizada sobre la mejora de la calidad del *software* a través de la integración y entrega continua muestra que esta práctica es altamente efectiva para mejorar la calidad del *software* y reducir el tiempo y los costos asociados con su desarrollo.

Los resultados de la investigación han demostrado que la integración y entrega continua pueden llevar a la detección temprana de errores, reducir los errores y problemas en el *software,* reducción del tiempo de corrección, aumentar la eficiencia y productividad de los desarrolladores, fomentar la colaboración y transparencia entre el equipo de desarrollo y mejorar la experiencia del usuario final.

Además, la entrega continua permite a las empresas lanzar nuevas versiones de *software* con mayor rapidez y frecuencia, lo que les permite mantenerse a la vanguardia de la competencia y responder rápidamente a las necesidades y demandas del mercado.

Referencias

Anderson, D. (2010). *Kanban: successful evolutionary change for your technology business.* Blue Hole Press.

Beck, K., Beedle, M., Bennekum, A., Cockburn, A., Cunningham, W., Fowler, M., & Thomas, D. (2001). *Manifesto for agile software development.*

Fowler, M. (01 de mayo del 2006). *Continuous integration.* https://martinfowler.com/articles/continuousIntegration.html

Hossain, M. S., & Muhammad, G. (2016). Cloud-assisted industrial internet of things (iiot)–enabled framework for health monitoring. *Computer Networks, 101*, 192-202. https://doi.org/10.1016/j.comnet.2016.01.009

Humble, J., & Farley, D. (2011). *Continuous delivery: reliable software releases through build, test, and deployment automation.* Pearson Education.

Kim, G., Humble, J., Debois, P., Willis, J., & Forsgren, N. (2016). *The DevOps Handbook: How to Create World-Class Agility. Reliability, and Security in Technology Organizations.*

Martin, R. (2009). *Clean code: a handbook of agile software craftsmanship.* Pearson Education.

Pressman, R. (2014). *Ingeniería del software: un enfoque práctico.* McGraw Hill.

Rossel, S. (2017). *Continuous Integration, Delivery, and Deployment: Reliable and faster software releases with automating builds, tests, and deployment*. Packt Publishing Ltd.

Schwaber, K., & Sutherland, J. (2017). The Scrum guide. *Scrum.org.* https://www.scrum.org/resources/scrum-guide